Engineering Data Analysis with MATLAB

Anke Scherb

Engineering Risk Analysis Group
Technische Universität München

Phone: 089 289 23013
E-mail: anke.scherb@tum.de

Engineering Data Analysis with MATLAB

Elective course– 6th semester BSc
2 ECTS

Lecture period: Tuesdays 25.04. – 16.05.2017
Lecture time: 08:45-10:15 am
Tutorials: 10:15-11:45 am
Lecture room: N 1039
Course language: English

- Register for the course in TUMonline – maximum number of participants 20
- Download MATLAB at https://matlab.rbg.tum.de/
  Using your myTUM credentials
  Guidance for installation is provided there
- Please bring your laptops with MATLAB installed

Lecture - contents

- Introduction to MATLAB: Basic commands
- Working with vectors/matrices
- Basic programming tools

- (Descriptive) Statistics of data sets
- Graphical representation of data sets
- Statistics of pairs of data sets
- Distributions and simulation of random variables

Exercises - contents

- Practical exercises to be solved in MATLAB

Examination: Project work

- Groups of 2 students
- Supervised by a tutor
- Duration of project work: 6 weeks
- Students must hand in a MATLAB code and a written report
- Attendance of 3 out of 4 tutorials is required for participation

Literature

Lecture Notes in Engineering Risk Analysis
Part B – Elementary Data Analysis
Prof. Dr. Daniel Straub
TU München

Lecture Notes in Engineering Risk Analysis
Part Z – Annex – MATLAB
Prof. Dr. Daniel Straub
TU München

Probability Concepts in Engineering: Emphasis on Applications to Civil and Environmental Engineering
A.H.-S. Ang and W.H. Tang
Wiley, 2006

Statistics, Probability and Reliability for Civil and Environmental Engineers
N.T. Kottegoda and R. Rosso
McGraw-Hill, 1997

http://www.mathworks.de/help/techdoc/

Introduction to MATLAB

Why do we need programming at all?

MATLAB = **MAT**rix **LAB**oratory – developed by MathWorks

- Proprietary software
- Main purpose: solve mathematical problems and present results with graphical illustrations
- Mainly used for numerical calculations/ simulations, and data acquisition/ analysis
- Includes a number of useful toolboxes (e.g. statistics, optimization…)

Introduction to MATLAB

Advantages

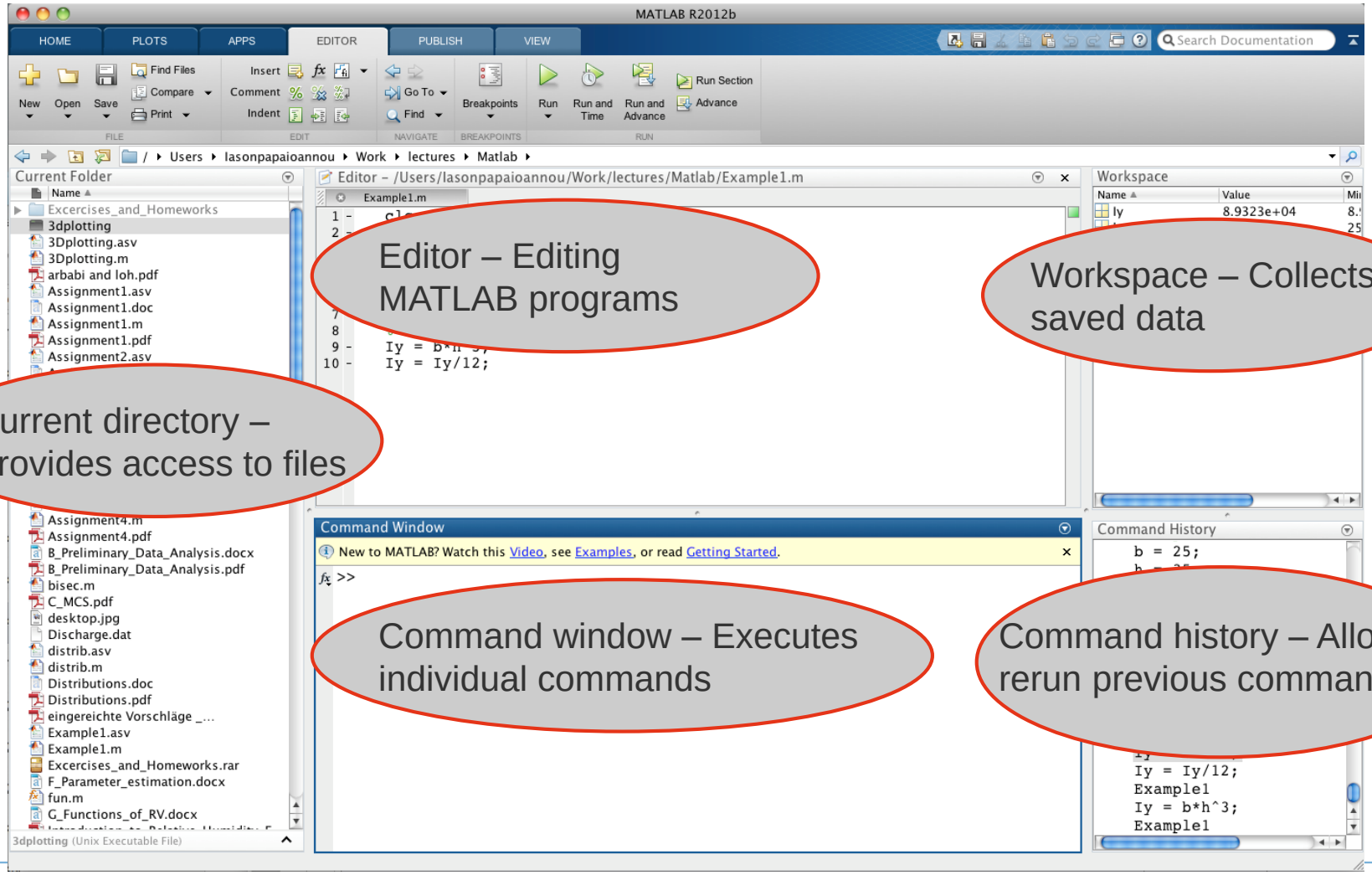
- Intuitive and easy to learn
- Fast on matrix operations
- Powerful syntax and great function library
- Good plotting options
- Easy debugging
- Additional toolboxes with functions for specific purposes

Disadvantages

- Interpreted language, no compiling
- Not designed for word processing, internet applications, GUIs
- Costs: commerically available

Free MATLAB alternative: octave

MATLAB desktop – working environment



MATLAB as calculator – A very first example

=> Example.calculator

Variables

Name convention:

- Start with letter
- Letters, numbers, underscores
- Case sensitive
- Use meaningful names!

Assigning values to variables

Variable name = expression.

Expression may be:

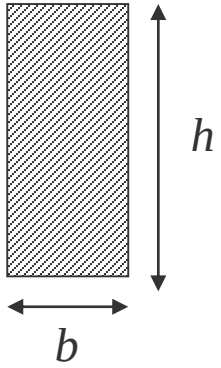
- Numerical value
- Values combined with operators
- Values of other variables
- Returns from MATLAB functions
- Returns from selfmade functions

Manual change in Workspace (go to window Workspace and change value)

MATLAB as calculator – A first example

Task: Compute the moment of inertia of a rectangular beam section

$$I_y = \frac{bh^3}{12}$$




```
1  clc
2  clear all
3
4  % Input dimensions of beam in cm
5  b = 25;
6  h = 40;
7
8  % Compute moment of inertia in cm^4
9  Iy = b*h^3;
10 Iy = Iy/12;
```

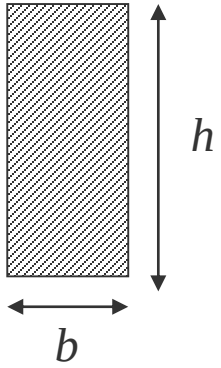
MATLAB as calculator – A first example

Task: Compute the moment of inertia of a rectangular beam section

$$I_y = \frac{bh^3}{12}$$

Clears command window

```
1  clc
2  clear all
3
4  % Input dimensions of beam in cm
5  b = 25;
6  h = 40;
7
8  % Compute moment of inertia in cm^4
9  Iy = b*h^3;
10 Iy = Iy/12;
```



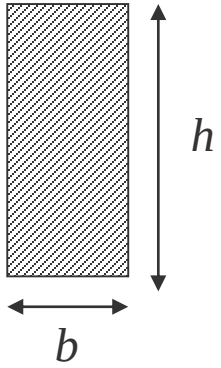
MATLAB as calculator – A first example

Task: Compute the moment of inertia of a rectangular beam section

$$I_y = \frac{bh^3}{12}$$

Removes all variables from memory

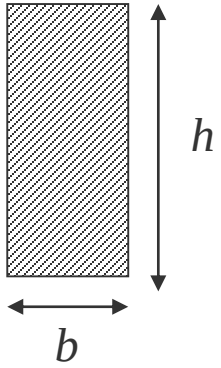
```
1  clc
2  clear all
3
4  % Input dimensions of beam in cm
5  b = 25;
6  h = 40;
7
8  % Compute moment of inertia in cm^4
9  Iy = b*h^3;
10 Iy = Iy/12;
```



MATLAB as calculator – A first example

Task: Compute the moment of inertia of a rectangular beam section

$$I_y = \frac{bh^3}{12}$$



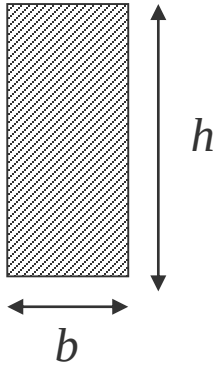

```
1   clc
2   clear all
3
4   % Input dimensions of beam in cm
5   b = 25;
6   h = 40;
7
8   % Compute moment of inertia in cm^4
9   Iy = b*h^3;
10  Iy = Iy/12;
```

Comment line

MATLAB as calculator – A first example

Task: Compute the moment of inertia of a rectangular beam section

$$I_y = \frac{bh^3}{12}$$




```
1   clc
2   clear all
3
4   % Input dimensions of be
5   b = 25;
6   h = 40;
7
8   % Compute moment of inertia in cm^4
9   Iy = b*h^3;
10  Iy = Iy/12;
```

Suppresses screen printing

Basic commands

`clc`	Clears the command window
`clear all`	Removes all variables from memory
`clear var1 var2`	Removes variables **`var1`** and **`var2`** from memory
`save mydata`	Saves data to **`mydata.mat`**
`load mydata`	Loads data from **`mydata.mat`**
`whos`	Lists variables in memory and their sizes
`;`	Suppresses screen printing
`...`	Continuous a line
`%`	Designates a comment line
`help fun`	Displays help topics about **`fun`**

Vectors

Row vector – Separate elements with spaces or commas

```
v = [3 1 7 -21];
```

Column vector – Separate elements with semicolons ‘;’

```
v = [3;1;7;-21];
```

Vectors

Vectors with regularly spaced elements – Colon sign ‘:’

```
v = [1:10]   % from 1 to 10 with increments of 1 (default)
v =
     1     2     3     4     5     6     7     8     9    10
```

```
v = [1:2:7]   % from 1 to 7 with increments of 2
v =
     1     3     5     7
```

```
v = [0.5:-0.1:-0.5] % from .5 to -.5; increments of -.1
v =
    0.5000    0.4000    0.3000    0.2000    0.1000
0   -0.1000   -0.2000   -0.3000   -0.4000   -0.5000
```

Vectors

Transpose operator – Quote sign ‘’’

```
>> v=[1:10]'      % Column vector
v =
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
```

Vectors

Access vector elements

```
v(1)
u(3)
w = v(1:2:4); % vector elements 1 to 4 with increments of 2
```

Vector functions

`length`	Computes number of elements
`linspace`	Creates regularly spaced vector

Example: Vectors.m

Matrices

Matrix – Column of row vectors or a row of column vectors

```
A = [1 2 3;4 5 6;7 8 9];
B = [[1;2;3] [4;5;6] [7;8;9]];
```

Access row or column of a matrix

```
u = A(1,:);  % access all column elements of row 1
v = A(:,2);  % access all row elements of column 2
```

Matrices

Matrix – Add columns to Matrices

```
>> A =  [1:3; 4:6; 7:9]  %  Or: A = reshape([1:9], 3, 3)

A =

     1     2     3
     4     5     6
     7     8     9
```

```
>> A=[A,[-1;-1;-1]]      %  Add column to Matrix A

A =

     1     2     3    -1
     4     5     6    -1
     7     8     9    -1
```

Matrices

Matrix – Add rows to Matrices/ Change Elements

```
>> A = [A;[-1,-1,-1,-1]]          % add row
A =
     1     2     3    -1
     4     5     6    -1
     7     8     9    -1
    -1    -1    -1    -1
```

```
>> A(3,3) = 10                    % change element
A =
     1     2     3    -1
     4     5     6    -1
     7     8    10    -1
    -1    -1    -1    -1
```

Matrices

Matrix – Change rows or columns

```
>> A(4,:) = [1, 1, 1, 1]  % change last row - Equivalent: A(4,:)=1
A =
     1     2     3    -1
     4     5     6    -1
     7     8    10    -1
     1     1     1     1
```

```
>> A(:,4) = [1; 1; 1; 1]  % change last column
A =
     1     2     3     1
     4     5     6     1
     7     8    10     1
     1     1     1     1
```

Matrices

Matrix – Delete rows or columns

```
>> A(4,:) = []              % delete last row
A =
     1     2     3    -1
     4     5     6    -1
     7     8    10    -1
```

```
>> A(:,4) = []              % delete last column
A =
     1     2     3
     4     5     6
     7     8    10
```

Matrices

Matrix operations – All mathematical symbols perform matrix operations

```
A+B   % Adds A and B
A-B   % Subtracts B from A
A*B   % Matrix multiplication
A/B   % A*inv(B)
```

Element-wise operators – Put a dot in front of a mathematical symbol to perform operations on individual elements of the matrices. Matrices must be of same size

```
A.*B
A./B
A.^B
```

Matrices

Math functions of matrices – Return matrices of same size with each entry specified by performing the operation at the corresponding entry of the original matrix

```
sin(A)

log(A)
```

Example: Matrices.m

Matrices

Matrix functions - built-in functions

`size`	Returns the size of each dimension of matrix
`max`	Returns largest element of each column
`min`	Returns smallest element of each column
`sort`	Sorts each column
`sum`	Sums each column
`inv`	Computes the inverse of a matrix
`'`	Computes the transpose of a matrix

Matrices

Special matrices: Identity Matrix

```
>> eye(3)                    % creates a 3x3 matrix
ans =
     1     0     0
     0     1     0
     0     0     1

>> eye(2,3)                  % creates a 2x3 matrix
ans =
     1     0     0
     0     1     0
```

Matrices

Special matrices: Zeros Matrix

```
>> zeros(3)                    % creates a 3x3 matrix
ans =
     0     0     0
     0     0     0
     0     0     0

>> zeros(2,3)                  % creates a 2x3 matrix
ans =
     0     0     0
     0     0     0
```

Matrices

Special matrices: Ones Matrix

```
>> ones(3)                    % creates a 3x3 matrix
ans =
     1     1     1
     1     1     1
     1     1     1

>> ones(2,3)                  % creates a 2x3 matrix
ans =
     1     1     1
     1     1     1
```

Saving the Workspace and loading data files

```
>> whos  % whos displays in alphabetical order all variables in
the currently active workspace


>> save ('myWS');          % saves workspace to myWS.mat


>> load('myWS.mat');       % laods data of myWS.mat file;
```

MATLAB – script file versus command window

- All of the pre-built commands that you use in MATLAB are script files or functions (plot, mean, std, exp, cosd, …)
- MATLAB allows the user to create his/her own customized m-files for specific applications or problems.
- A script file is simply a collection of executable MATLAB commands. To create a new script file, click on the New Script icon on the left side of the Home Tab.
- Save the file in an appropriate folder. When you pick a name for the file you must follow the same rules that MATLAB has for naming variables.
- Set the current directory in MATLAB to the same place where you saved the script file.
- To run the script file: Hit the Green Run Arrow in the Editor window

Example: CircleScript.m

Elementary Data Analysis – Descriptive statistics

- All natural and human processes are subject to variability of some kind
  => hence there is most often uncertainty to some extent

- **Goal:** Characterization and quantification of uncertainty to understand the process

- Descriptive statistics (data analysis)

- Graphical representation of the data

- **Goal:** Choice of appropriate probabilistic distributions and their parameters for estalishment of probabilistic engineering models

Elementary Data Analysis – Descriptive statistics

- Collection of experimental (measured) data/ samples

- E.g. laboratory tests to determine water quality parameters, properties of engineering materials, soil characteristics …

Example: Tensile strength and Young's modulus from tests on timber specimens (provided by Lehrstuhl für Holzbau und Baukonstruktion, TUM).

Specimen #	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17
Tensile strength [N/mm²]	37.1	36.9	40.5	46.8	37.6	30.0	29.0	27.5	30.7	43.9	29.2	28.7	18.3	34.3	35.0	58.0	20.9
Young's modulus [N/mm²]	10237	6951	12242	20814	18815	13005	9886	8155	11437	11718	8348	9368	10123	11677	10770	9993	8808
Specimen #	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
Tensile strength [N/mm²]	26.8	18.4	34.0	28.4	35.2	27.1	38.0	28.0	30.3	54.9	55.4	32.8	39.5	19.3	54.2	23.1	50.3
Young's modulus [N/mm²]	11598	9530	10669	12648	11765	12076	12950	8799	7841	12568	11830	11676	9576	8364	12574	6611	12399
Specimen #	35	36	37	38	39	40	41	42	43	44	45	46	47	48	49	50	
Tensile strength [N/mm²]	66.5	32.7	35.7	36.2	26.4	54.8	27.5	25.3	48.3	47.7	52.9	29.9	42.4	24.8	28.9	26.0	
Young's modulus [N/mm²]	14024	12176	10796	11384	10330	12723	12608	7956	12886	10771	11158	9223	10457	8278	8587	9928	

Numerical summaries of data sets

Measures of central tendencies

Sample range – Interval between the lowest and the largest value of the data set

```
range(data)
```

Sample mean – arithmetic mean of the n sample values x_i

$$\bar{x} = \frac{1}{n}\sum_{i=1}^{n} x_i$$

```
mean(data)
```

Note: The sample mean is sensitive to outliners (unexpectedly low or high values)

Numerical summaries of data sets

Measures of central tendencies

Sample median – The value for which there is an equal number of larger and smaller samples

- Sort data in ascending order
- If the number of samples n is odd then the median is equal to the $(n + 1)/2$ th sample
- If n is even then the median is obtained as the mean of the $n/2$ th and the $(n/2 + 1)$ th sample

```
median(data)
```

Note: The sample median is not affected by outliners

Numerical summaries of data sets

Measures of dispersion (range, variance, standard deviation, interquantile range)

Sample variance – Mean of the square of differences between the samples and the sample mean

$$s^2 = \frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2$$

```
var(data)
```

Note: The denominator $n-1$ leads to an unbiased estimator of the variance

Numerical summaries of data sets

Measures of dispersion (range, variance, standard deviation, interquantile range)

Sample standard deviation – Square root of the sample variance

$$s = \sqrt{\frac{1}{n-1}\sum_{i=1}^{n}\left(x_i - \bar{x}\right)^2}$$

```
std(data)
```

Numerical summaries of data sets

Sample quantiles – Values that separate the ordered samples into k groups of equal number of samples. E.g. quartiles divide the samples into four groups of $n/4$ samples.

Sample percentiles – Values below which a certain percentage of all samples lie. E.g. the 10-percentile x_{10} is the value below which 10% of samples lie

```
quantile(data,p)
```

```
prctile(data,p)
```